to be able to compute a finite function ϕ.

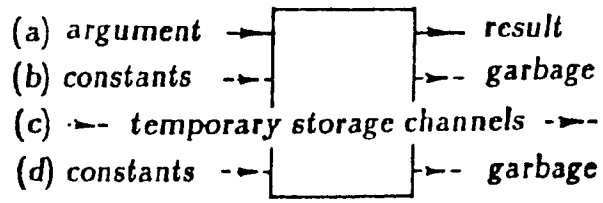

FIG. 5.5 *Classification of input and output lines in a reversible combinational network, according to their function. (a) Argument and result of the intended computation. (b) Constant and garbage lines to account for the noninvertibility of the given function. (c) "Temporary storage" registers required when only a restricted set of primitives is available. (d) Additional constant and garbage lines required when in designing the network one chooses not to take full advantage of the correlation between internal streams of data, and thus looses opportunities to bring about destructive interference of garbage.*

(a) If ϕ is invertible, and the use of *ad hoc* primitives is allowed, then no source or sink lines are necessary.

(b) If ϕ is not invertible, it is necessary to provide source lines to be fed with specified constants and/or sink lines which will produce garbage.

(c) Independently of the invertibility of ϕ, if only a restricted set of primitives is allowed it may be necessary to supply temporary-storage channels (i.e, additional source lines to be fed with constants and an equal number of sink lines which will return constants).

(d) If, in order to simplify the mechanism, one chooses to ignore that certain streams of garbage within the mechanism are correlated and thus could be subjected to destructive interference, then further source and sink lines will be required, as in (b).

6. Conservative logic

In view of the considerable exertions that were necessary in the previous sections in order to obtain a bona fide realization of mechanisms having universal logic capabilities and at the same time satisfying the reversibility constraint, one might wonder whether nontrivial computation would be possible at all if substantial further constraints were imposed.

Actually, it turns out that universal logic capabilities can still be obtained even if one restricts one's attention to combinational networks that, in addition

to being reversible, conserve in the output the number of 0's and 1's that are present at the input. The study of such networks is part of a discipline called *conservative logic** [7] (also cf. [11]). As a matter of fact, most of the results of Sections 4 and 5 were originally derived by Fredkin and associates in the context of conservative logic.

In conservative logic, all data processing is ultimately reduced to *conditional routing* of signals. Roughly speaking, signals are treated as unalterable objects that can be moved around in the course of a computation but never created or destroyed. The physical significance of this principle will be discussed shortly.

The basic primitive of conservative logic is the *Fredkin gate* (cf. [76]), defined by the table

$$\begin{array}{ccc} c & x_1 & x_2 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array} \rightarrow \begin{array}{ccc} c' & y_1 & y_2 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array} . \tag{6.1}$$

This computing element can be visualized as a device that performs conditional crossover of two *data* signals a and b according to the value of a *control* signal c (Figure 6.1a). When $c = 1$ the two data signals follow parallel paths, while when $c = 0$ they cross over (Figure 6.1b).

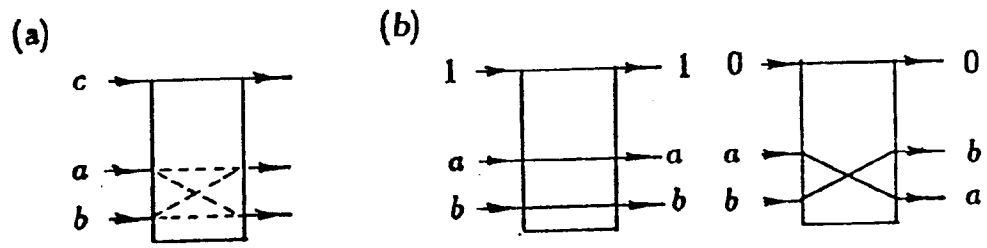

FIG. 6.1 *(a) Symbol and (b) operation of the Fredkin gate.*

In order to prove the universality of this gate as a logic primitive for reversible computing, it is sufficient to observe that AND can be obtained from the mapping

*Besides a combinational primitive (the *Fredkin gate* discussed below), conservative logic also posits a communication/memory primitive (the *unit wire*). Together, these two primitives are sufficient to account for sequential computation. Moreover, they provide a basis for a theory of computation complexity which incorporates certain important physical-like constraints that are usually neglected.

$\langle p, q, 0\rangle \mapsto \langle p, pq, \bar{p}q\rangle$, and NOT and FAN-OUT from the mapping $\langle p, 1, 0\rangle \mapsto \langle p, p, \bar{p}\rangle$.

There exists only one other 3-input, 3-output conservative-logic element that can be used as a universal primitive; this is the *symmetric majority/parity gate* (SMP) defined by

$$\begin{array}{ccc} x & y & z \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array} \rightarrow \begin{array}{ccc} x' & y' & z' \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array} . \tag{6.2}$$

Intuitively, in the SMP gate the three signals are rotated in one direction if their overall parity is even, and in the other direction if it is odd. While the SMP gate can be realized isomorphically by means of Fredkin gates, a realization of the latter by the former requires a temporary-storage channel. In this sense, the Fredkin gate is the *most elementary* conservative-logic primitive—since no 2-input, 2-output invertible function is universal.*

Finally, the Fredkin gate can be realized *isomorphically* by means of AND/NAND gates, as shown in Figure 6.2.

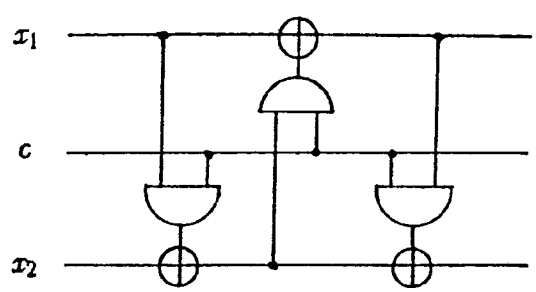

FIG. 6.2 *Isomorphic realization of the Fredkin gate by means of* AND/NAND *gates.*

It is important to realize that, far from representing merely an elegant *tour-de-force* in the theory of reversible computing, conservative logic provides an essential connection between that theory and the physics of computing circuits. In

*An even more elementary conservative-logic primitive, namely, the *interaction gate*[7], can be obtained if one considers invertible function whose domain and codomain coincide with a *proper subset* of a set of the form $\mathbf{B}^n$.

fact, while in any reversible system whatsoever there are a number of conserved quantities, in *physical* system many of these quantities are required to have a special structure. For instance, energy, momemtum, and angular momentum are *additive* (this reflects certain symmetries of space and time). Intuitively, reversibility by itself is not sufficient to give enough physical "flavor" to a theory of computing.

In a conservative logic circuit, the number of 1's, which is conserved in the operation of the circuit, is the sum of the number of 1's in different parts of the circuit. Thus, this quantity is additive, and can be shown to play a formal role analogous to that of energy in physical systems. Other connections between conservative logic and physics will discussed in more detail in [7]

In conclusion, conservative logic represents a substantial step in the development of a model of computation that adequately reflects the basic laws of physics.

7. Reversible sequential computing

In Sections 4 and 5, we started from a certain computing object (viz., a finite function), and we discussed the conditions for its *reversible* realization first (*a*) as an object of the same nature (viz., an invertible finite function) treated as a "lumped" system, thus stressing *functional* aspects, and then (*b*) as a "distributed" system (viz., a reversible combinational network), thus stressing *structural* aspects and paving the way for a natural physical implementation.

By and large, we shall follow a similar plan in dealing with the more complex computing objects that constitute the paradigms of sequential computing, namely, *finite automata* (in the present section), *Turing machines* (Section 8), and *cellular automata* (Section 9).

First, we shall make a few commemts on sequential computing in general.

In Section 2, we defined an abstract computer as a function-composition scheme. Some of these schemes possess a regular structure of a certain kind, namely, they are *time-iterative*,* and with appropriate conventions can be represented much more compactly in terms of *sequential* networks (cf. [9]).

*If the *partial-order* relation between arcs that is induced by the function-composition operation is interpreted as a *causal* relationship between signals, then certain other relations between arcs are naturally interpreted as referring to *spatial* and *temporal* relationships. In this context, a causal network may possess certain automorphisms that may be interpreted as *time shifts*, and in this case the network is *time-iterative*; similarly, it may possess automorphisms corresponding to *spatial* shifts, and in this case it is *space-iterative*.

Consider, for definiteness, the iterative function-composition scheme

$$\begin{cases} q_{i+1} = x_i \oplus q_i \\ y_{i+1} = x_i \oplus q_i \end{cases} \quad (-\infty < i < +\infty), \tag{7.1}$$

explicitly represented by the infinite causal network of Figure 7.1.

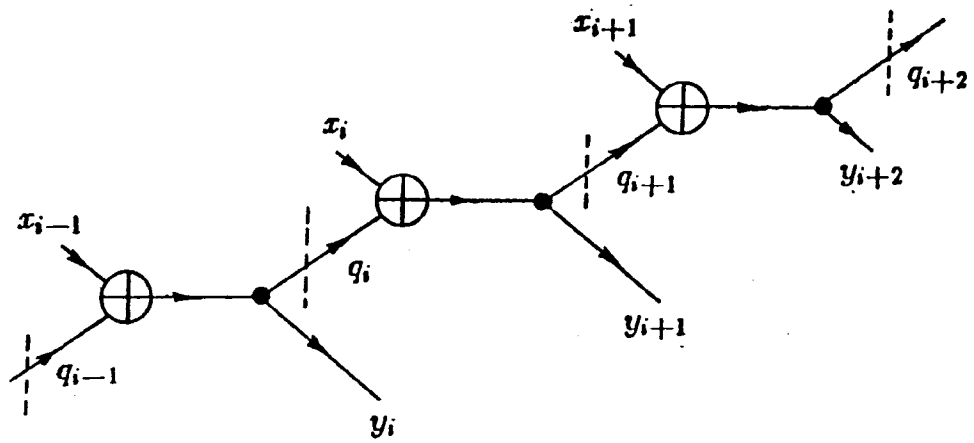

FIG. 7.1 *A function-composition scheme having a time-iterative structure.*

The same scheme can be represented by the finite sequential network of Figure 7.2. By comparing this figure with Figure 7.1, it is clear that the role of the so-called "delay elements" in a sequential network is to mark out those places that correspond to the boundary between one stage and the next in the equivalent time-iterative combinational network.

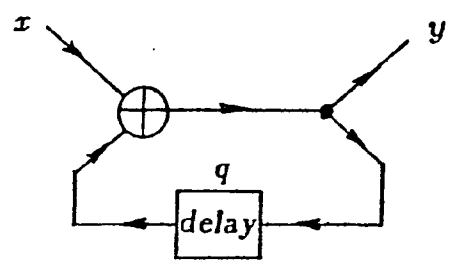

FIG. 7.2 *The same function-composition scheme as represented by a finite sequential network.*

In the previous sections, for the sake of realization arguments we have treated a finite function as a combinational network consisting of a single node. Similarly, we shall treat a finite automaton as a sequential network whose combinational part consists of a single node, representing the automaton's transition function. Thus, for example, the network of Figure 7.2 can be identified with the finite automaton $\langle x, q\rangle \mapsto \langle x \oplus q, x \oplus q\rangle$.

Finally, recalling that a computation is a solution of a function-composition scheme (cf. Section 2), let us note that one is usually not interested in arbitrary solutions; rather, one seeks solutions corresponding to particular boundary conditions (defined, for instance, by assigning specified values to the input lines). Moreover, in the case of a time-iterative scheme, one is usually interested only in that portion of the scheme whose elements are indexed by $i \geq i_0$ (for an arbitrary i_0). The assignment of specified values to the new input lines that are created in this way (i.e., by "cutting" the network in half) corresponds to *initializing* the delay elements in the sequential-network representation.

By definition, a finite automaton is *reversible* if its transition function is invertible. Thus, in order to realize a finite automaton by means of a reversible sequential network, it will be sufficient to take its transition function, construct a reversible realization of it, and use this as the combinational part of the desired sequential network. The problem of reversibly realizing an arbitrary finite function has been solved in Section 4. Thus, we have the following theorem.

THEOREM 7.1. *For every finite automaton $\tau: X \times Q \to Q \times Y$, where $X = B^m$, $Y = B^n$, and $Q = B^u$, there exists a reversible finite automaton $t: (B^r \times B^m) \times B^u \to B^u \times (B^n \times B^{r+m-n})$, with $r \leq n + u$, such that*

$$t_i(\overbrace{0, \ldots, 0}^{r}, x_1, \ldots, x_m, q_1, \ldots, q_u) = \tau_i(x_1, \ldots, x_m, q_1, \ldots, q_u), \quad (i = 1, \ldots, u+n).$$

In other words, whatever can be computed by an arbitrary finite automaton according to the scheme of Figure 7.3a can also be computed by a *reversible* finite automaton according to the schema of Figure 7.3*b*.

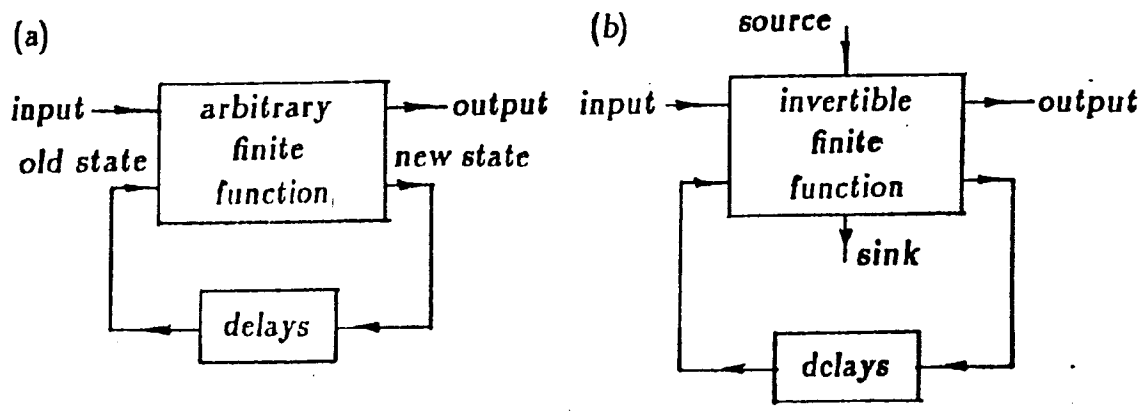

FIG. 7.3 *Any finite automaton (a) can be realized as a reversible finite automaton (b) having a number of auxiliary input lines (source) which are fed with constants and a number of auxiliary output lines (sink) whose values are disregarded.*

The following theorem, which from a mathematical viewpoint is trivial, is a restatement of the second law of thermodynamics. Recall that an automaton is *closed* if $X = \{\lambda\}$ (i.e., there are no input lines); *open*, otherwise.

THEOREM 7.2 *If a closed, finite automaton is not reversible, then it does not admit of a reversible realization that is both closed and finite.*

Proof. Given a nonreversible automaton $\tau : Q \to Q$, assume that $t : P \to P$ be a reversible realization of τ. That is, we assume the existence of an invertible finite function t and of two maps μ and ν (respectively, the *encoder* and the *decoder*) such that, for all $i \geq 0$,

$$\nu t^i \mu = \tau^i. \tag{7.2}$$

Since automaton t is finite and reversible, for any $p \in P$ there exists an $i_p > 0$ such that $t^{i_p}(p) = p$. Thus, for any $q \in Q$, $\nu t^{i_p(q)}\mu(q) = \nu\mu(q)$. But, in view of (7.2), $\nu\mu(q) = q$ and $\nu t^{i_p(q)}\mu(q) = \tau^{i_p(q)}(q)$. As a consequence, $\tau^{i_p(q)}(q) = q$ for all q. On the other hand, since automaton τ is finite and nonreversible, there must be at least one $\bar{q} \in Q$ such that $\tau^i(\bar{q}) \neq \bar{q}$ for all $i > 0$. Thus, the original assumption leads to a contradiction.∎

Remark. According to Theorem 4.1, noninvertible functions can be computed by a reversible system provided that a supply of constants is made available. In an open system, this supply may come from some input lines. In a

closed system, in order to perform a computation in one part of the system one may draw constants from another part, for instance, the indefinitely extended blank tape of a Turing machine (cf. Section 8). However, if the system is also finite, this supply will eventually run out.

Thus, the interpretation that a finite, closed, and reversible system "is modeling an irreversible process" can only be maintained for particular initial conditions and on a space-time scale that is limited with respect to that of the entire system.

Having discussed the realization of finite automata by means of reversible finite automata, we turn now to the realization of finite automata by means of reversible finite sequential networks based on given primitives.

It is clear that all the arguments of Sections 5 and 6 concerning finite functions immediately apply to the transition function of any given finite automaton. In particular, in analogy with Figure 5.5, every finite automaton can be realized by a finite, reversible sequential network based on, say, the AND/NAND primitive and having the following form (Figure 7.4)

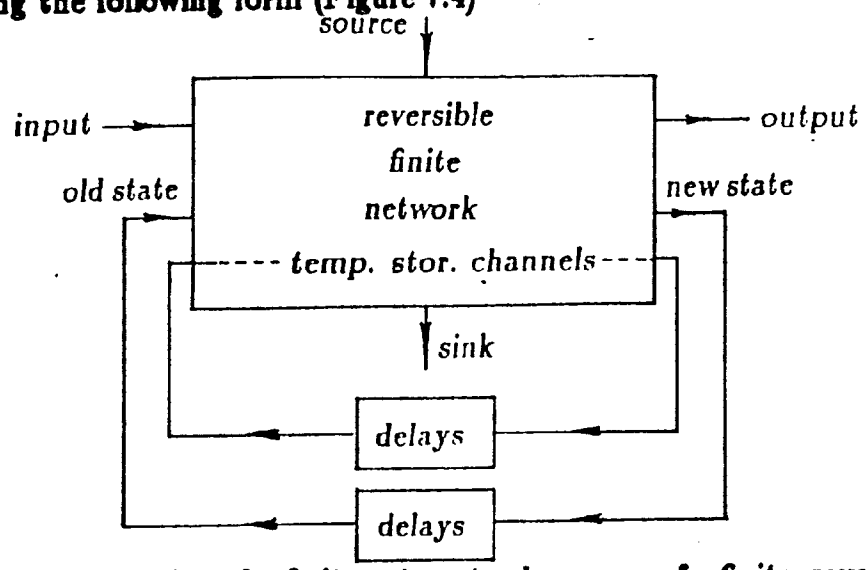

FIG. 7.4 *Realization of a finite automaton by means of a finite, reversible sequential network.*

In order to insure the desired behavior, the state components represented by the temporary-storage channels must be initialized *once and for all* with appropriate values (typically, all 0's), while the source must be fed with appropriate constants (typically, all 0's) *at every sequential step.*

In a conventional computer, power dissipation is proportional to the number of logic gates. On the other hand, the number of constants/garbage lines in

Figure 7.4 is at worst proportional to the number of input/output lines (cf. Theorems 4.1 and 5.3). From the viewpoint of a physical implementation, where signals are encoded in some form of energy, the above schema can be interpreted as follows: *Using invertible logic gates, it is ideally possible to build a sequential computer with zero internal power dissipation.* The only source of power dissipation arises outside the circuit, typically at the input/output interface, if the user chooses to connect input or output lines to nonreversible digital circuitry. Even in this case, power dissipation is at most proportional to the number of argument/result lines,* rather than to the number of logic gates (as in ordinary computers), and is thus independent of the "complexity" of the function being computed. This constitutes the central result of the present paper.

8. Reversible Turing machines

We shall assume the reader to be familiar with the concept of *Turing machine*. From our viewpoint, a Turing machine is a closed, time-discrete dynamical system having three state components, i.e., (*a*) an infinite *tape*, (*b*) the internal state of a finite automaton called *head*, and (*c*) a *counter* whose content indicates on which tape square the head will operate next. Let T, H, and C be the sets of tape, head, and counter states, respectively. A Turing machine is *reversible* if its transition function $\tau: T \times H \times C \rightarrow T \times H \times C$ is invertible.

It is well known that for every recursive function there exists a Turing machine that computes it, and, in particular, that there exist computation-universal Turing machines. Are these capabilities preserved if one restricts one's attention to the class of *reversible* Turing machines?

The answer to the above question is positive. In fact, in [4] Bennett exhibits a procedure for constructing, for *any* Turing machine and for certain quite general computation formats, a reversible Turing machine that performs essentially the same computations. (The concept of "reversibility" used by Bennett is weaker than ours, insofar as the transition function and its inverse are defined only on distinguished subsets of tape/head/counter states, and even among these states there are some with no predecessors and others with no successors; however, the transition rules of Bennett's machine can easily be augmented so as to satisfy

*In fact, the free energy that must be supplied is at most that in which the input constants are encoded, and the thermal energy that must be removed is at most that of the garbage outputs. According to Theorem 4.1, the number of constant lines need not be greater than that of result lines, and the number of garbage lines need not be greater than that of argument lines.

our stronger definition. We shall assume this augmenting procedure to have been carried out.)

In order to obtain the desired behavior, Bennett's machine is initialized so that all of the tape is blank except for one connected portion representing the computation's argument, and the head is set to a distinguished "initial" state and positioned by the argument's first symbol. At the end of the computation, i.e., when the head enters a distinguished "terminal" state, the result will appear on the tape alongside with the argument, and the rest of the tape will be blank.

Thus, a number of tape squares that are initially blank will eventually contain the result. These squares fulfill a role similar to that played by the constants/garbage lines in Section 5, in the sense that they provide a sufficient supply of "predictable" input values (blanks) at the beginning of the computation, and collect the required amount of "random" output values (in this case, a copy of the argument—cf. the first row of (4.2)) at the end of the computation.

Moreover, during the computation a number of originally blank tape squares may be written over and eventually erased. These squares fulfill a role similar to that played by the temporary-storage lines in Section 5 (cf. the discussion centered on Figure 5.4).

It is clear that, like the constants in the reversible combinational networks of Section 5, the blanks in Bennett's machine play an essential role in the computation, since without their presence one could not achieve *universality* and *reversibility* at the same time. Intuitively, computation in reversible systems requires a *higher degree of "predictability" about the environment's initial conditions* than computation in nonreversible ones.

Owing to the "hybrid" nature of a Turing machine, which involves several kinds of computing primitives (an active device, viz., the head, a passive device, viz., the tape, and mechanisms for head-tape interaction and for head movement), it is difficult to discuss the realizability of a Turing machine by means of a reversible, distributed computing mechanism before having expressed the above primitives in terms of fewer and more elementary ones. For this reason, we shall deal with this problem at the end of the next section, when suitable tools will have been introduced.

9. Reversible cellular automata

We shall assume the reader to have some familiarity with the concept of *cellular automaton* (cf. [21] for details and references). From a physical viewpoint, cellular automata are in many respects more satisfactory models of computing

processes than Turing machines; in particular, they allow one to represent by means of the same mathematical machinery and at the same level of abstraction not only a computer proper but also its environment, the interface with its "users," and the "users" themselves[22]. A cellular automaton is in essence a k-dimensional ($k \geq 1$) array of identical, uniformly interconnected finite automata, and constitutes a closed, time-discrete dynamical system whose state set is a countable Cartesian product of finite sets and whose transition function is continuous (in the topological sense) and commutes with the translations (i.e., with the elements of the array's symmetry group). A cellular automaton is *reversible* if its transition function is invertible.

It is well known that there exist cellular automata that are computation- and construction-universal. Are these capabilities preserved if one restricts one's attention to the class of *reversible* cellular automata? The answer to the above question is positive. In fact, in [20] Toffoli exhibits a procedure for constructing, for *any* cellular automaton (presented as an infinite, space-iterative sequential network), a reversible cellular automaton that realizes it. (Note that until then the existence of computation- and construction-universal reversible cellular automata—which is thus established—had been doubted or, by erroneous arguments, outright denied.)

Assuming the original cellular automaton $\hat{M}$ to be *one-dimensional* and with the von Neumann neighborhood (Figure 9.1a), with Toffoli's construction one obtains a *two-dimensional* cellular automaton M in which the flow of information between the elements of the array (or *cells*) can be visualized as in Figure 9.1*b*.

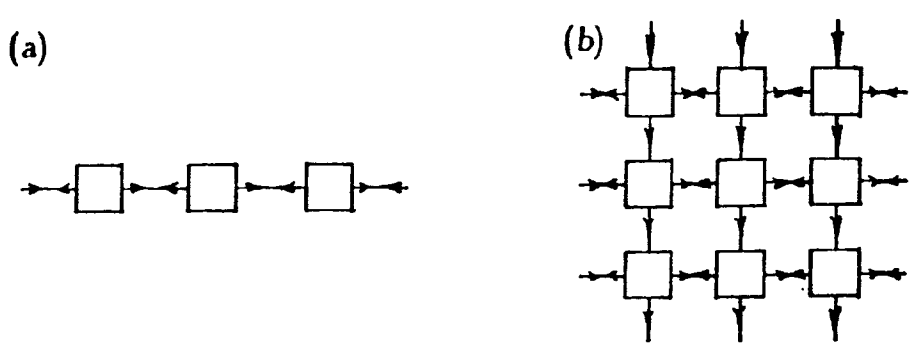

FIG. 9.1 *Information flow (a) in a given one-dimensional cellular automaton $\hat{M}$, and (b) in the corresponding two-dimensional, reversible realization M.*

Each row of M constitutes a reversible (infinite) automaton which is, of course, open. Let us consider a particular row ρ. In analogy with Figure 7.3, let us interpret the arrows entering a row from above (in Figure 9.1) as *source lines*,

and the ones leaving it from below as *sink* lines. By construction, when the source lines are fed with particular constant values (all 0's) and the values on the sink lines are ignored, the "row" automaton will behave as $\hat{M}$. In order for M to constitute a reversible realization of $\hat{M}$, one must guarantee that ρ's source lines are fed with 0's at all time steps $t \geq 0$. But, by construction, if a row is initialized to a distinguished *blank* state and its source lines are fed with 0's, then the row will remain in the blank state and its sink lines will produce 0's. Thus, if all of the rows that lie above ρ are initialized to the blank state (Figure 9.2), an inexhaustible supply of 0's will "rain down" into ρ. At the same time, ρ's sink lines will produce garbage that will spread down and sideways through the rows below ρ, while ρ itself will reproduce $\hat{M}$'s behavior.

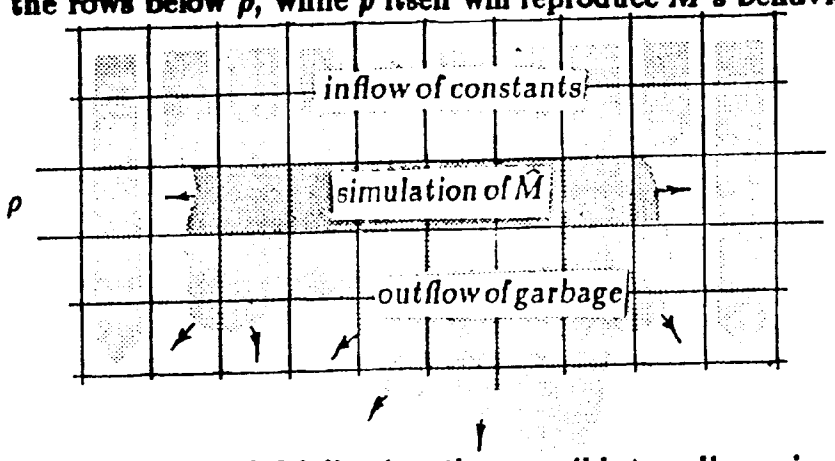

FIG. 9.2 *With proper initialization, the reversible two-dimensional cellular automaton M realizes the one-dimensional automaton $\hat{M}$, which may be irreversible.*

In Section 5, we saw that, starting from a fixed, finite set of invertible logic primitives, one cannot obtain an *isomorphic* invertible realization for every invertible finite function; rather, in general one must be allowed to use ad-hoc primitives (possibly as "complex" as the function itself) or introduce temporary-storage channels (thus giving up isomorphism). Similar considerations apply to finite automata.

Consider now a reversible cellular automaton $\hat{M}$. $\hat{M}$ can always be realized as a reversible sequential network *with temporary storage*. Is it possible to construct an *isomorphic* reversible realization of $\hat{M}$ if one is given a free hand in the choice of primitives? Surprisingly, the answer to this question is negative (Theorem 9.1 below).

LEMMA 9.1. *Let m be the number of binary state variables associated with each cell of a cellular automaton $\hat{M}$, and n the size of its neighborhood. A neces-*

sary condition for the isomorphic realizability of $\hat{M}$ as a reversible sequential network is that $m \geq n$.

Proof. We shall sketch the proof for a one-dimensional cellular automaton $\hat{M}$ with $m = 1$, $n = 2$, and neighborhood index $(0, -1)$ (i.e., the next state of cell i will depend on the current state of cell i itself and of cell $i - 1$, that is, its "left" neighbor . Assume that the required realization M exists; it will have the form of Figure 9.3a.

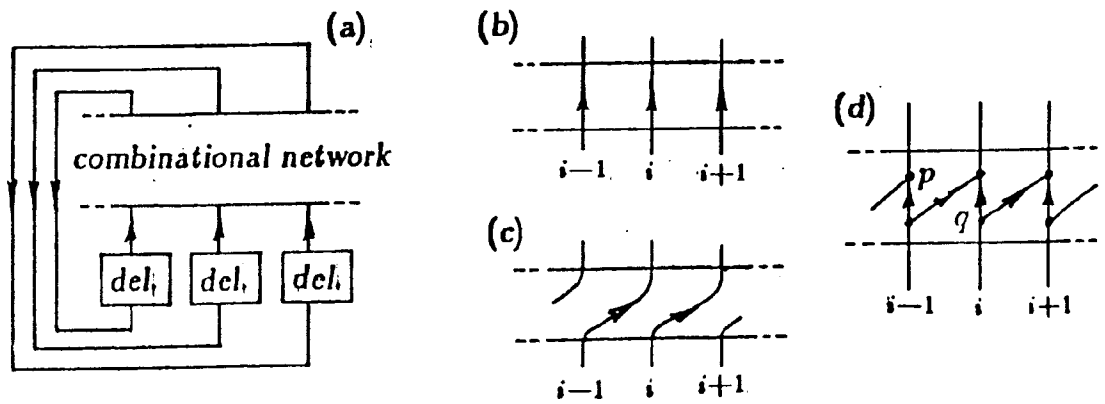

FIG. 9.3 *(a) Overall structure of the (assumed) isomorphic reversible realization of $\hat{M}$. Details of the combinational part of such realization: (b) paths from each cell to itself, (c) paths from each cell to its right neighbor, and (d) nodes implied by the existence of such paths.*

Since each cell is "affected" by itself, there will be paths in the combinational part of M as in Figure 9.3*b*; similarly, since each cell is "affected" by its left neighbor, there will be paths as in Figure 9.3*c*. Thus, there will be nodes as in Figure 9.3*d*. These nodes cannot stand for invertible primitives (note fan-in at p and fan-out at q) unless further arcs exist (namely, an additional arc entering p and one leaving q). In turn, these arcs imply the existence of new nodes—and so on *ad infinitum*—since connecting these arcs to one another would violate the partial ordering ("causality") associated with function composition). Clearly, it is impossible to complete the construction without introducing either cycles or paths of infinite length.∎

THEOREM 9.1. *There exist reversible cellular automata that do not admit of an isomorphic reversible realization.*

Proof. In [2], Amoroso and Patt exhibit reversible cellular automata (more

amply discussed in [21]) with $m = 1$ and $n = 4$. The thesis follows from Lemma 9.1.∎

Finally, it is well known that any Turing machine can be embedded in a suitable cellular automaton. Thus, according to the foregoing discussion, any Turing machine can be realized by an infinite *reversible* sequential nework.

10. Conclusions and perspectives

What properties of an abstract computing system are critical in determining whether or not it can carry out a given task? To give an example, it is known that *linearity* is incompatible with computation universality[15]. Many other issues, dealing, for instance, with monotonicity, finiteness of storage, availability of constants, etc., have been discussed extensively in the past and are essentially settled. On the other hand, only recently has some light been thrown on the question of to what extent the imposition of the *reversibility* constraint affects the computing capabilities of a system. The sustained debate on this issue has been fueled and at times confused by obvious connections with certain physical questions.

In this paper, we have laid down the foundations for the *theory of reversible computing*, i.e., a theory of computing based on invertible primitives and composition rules that preserve invertibility, and we have shown that this theory can deal satisfactorily with both the functional and the structural aspects of computing processes. In particular, those structures that constitute the traditional paradigms of the theory of computing, such as combinational and sequential networks, finite automata, Turing machines, and cellular automata, have been ascertained to have reversible counterparts of equal computing power. (Analogous considerations apply to systems of difference equations, as will be shown in a later paper.) Thus, the choice to use reversible mechanisms in describing computing processes is a viable one. What can be gained from this choice?

In the synthesis of an abstract computing system for a given task, the requirement that the system be reversible can in general be met only at the cost of greater structural complexity. In other words, one may need more (but not many more) gates and wires. However, the system's very reversibility promises to be a key factor in leading to a more efficient physical realization, since, at the microscopic level, the "primitives" and the "composition rules" available in the physical world resemble much more closely those used in the theory of reversible computing than those used in traditional logic design. For example, it appears that conservative logic might be completely modeled by processes of elastic collision between identical particles[7].

Besides questions of efficient implementation of computing systems, there are several other research lines to which the concepts of reversible computing may give a substantial contribution. Here, we shall limit ourselves to naming a few on which some preliminary investigations have been carried out, such as error detection; numerical analysis and, in particular, the integration of differential equations; number theory and the factoring of large integers; second-order automata and state-reduction techniques; synchronization of computing processes; and cellular automata and the modeling of fundamental physical laws.

The main reason why the concept of reversibility is so pervasive is the following. In every reversible system there are a number of quantities (functions of the system's state) which are *conserved*, i.e., which are constant along each of the system's trajectories (cf. the "integrals of the motion" of theoretical mechanics). Both in mathematical and in physical research, experience has shown that many interesting properties of a system can usually be expressed in terms of conserved quantities, and on this basis one can often make nontrivial statements about the behavior of a system or a class of systems without having to go into a detailed, case-by-case analysis of their operation.

Thus, in the more abstract context of computing, the laws of "conservation of information" may play a role analogous to those of conservation of energy and momentum in physics.

11. Historical and bibliographical notes

The computing element known as the "Fredkin gate" was apparently first described by Quine as an abstract primitive, and was known to Petri[10]. Some conservative (but not reversible) circuits using complementary signal streams were discussed by von Neumann[25] as early as 1952. More recently, Kinoshita and associates[11] worked out a classification of logic elements that "conserve" 0's and 1's; their work, motivated by research in magnetic-bubble circuitry, mentions the possibility of more energy-efficient computation, but has apparently little concern for reversibility.

The AND/NAND element was known to Petri[10], and was mentioned *en passant*, in the context of reversibility arguments, by Landauer[12]. The generalized AND/NAND functions were introduced by Toffoli[20] for proving the universality of reversible cellular automata.

Doubts about the computational capabilities of reversible cellular automata were first expressed by Moore[16], and more formally stated as a conjecture by Burks[5]. Other parties in the debate were Smith[18], Amoroso and Patt[2], Aladyev[1], and Di Gregorio and Trautteur[6].

Distributed systems which are not reversible *per se*, but in which computation-universal, reversible structures can be embedded by proper initialization, were described by Priese[17].

The idea that universal computing capabilities could be obtained from reversible, dissipationless (and, of course, nonlinear) physical circuits apparently first occurred to von Neumann, as reported in a posthumous paper[26]. This idea was also aired by Bennett[4], with some support from his construction of reversible Turing machines. A more formal proof of the physical realizability of such circuits was given by Toffoli[23], and some concrete though not yet entirely practical approaches are outlined in [8]. A much more promising approach, based on the interaction gate, has been recently suggested by Fredkin[7].

Owing to the interdisciplinary nature of the subject, it is quite difficult to locate, evaluate, and credit all research results that may somehow be relevant to reversible computing. We shall appreciate any contributions to the above bibliographical notes.

Acknowledgments

As acknowledged in the text, many ideas discussed in the present paper were originated by Prof. Edward Fredkin. Aside from this, I wish to thank him for much useful advice and encouragment.

List of references

[1] ALADYEV, Viktor, "Some Questions Concerning Nonconstructibility and Computability in Homogeneous Structures," *Izv. Akad. Nauk. Estonian SSR*, Fiz.-Mat. **21** (1972), 210-214.

[2] AMOROSO, Serafino, and PATT, Y. N., "Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures," *J. Comptr. Syst. Sci.* **6** (1972), 448-464.

[3] BARTON, Edward, "A Reversible Computer Using Conservative Logic," *6.895 Term Paper*, MIT Dept. of Electr. Eng. Comp. Sci. (1978).

[4] BENNETT, C. H., "Logical Reversibility of Computation," *IBM J. Res. Dev.* **6** (1973), 525-532.

[5] BURKS, A. W., "On Backwards-Deterministic, Erasable, and Garden-of-Eden Automata," *Tech. Rep. no. 012520-4-T*, Comp. Comm. Sci. Dept., Univ. of Michigan (1971).

[6] Di Gregorio, Salvatore, and Trautteur, Giorgio, "On Reversibility in Cellular Automata," *J. Comptr. Syst. Sci.* **11** (1975), 382-391.

[7] Fredkin, Edward, and Toffoli, Tommaso, "Conservative Logic," (in preparation). Some of the material of this paper is tentatively available in the form of unpublished notes from Prof. Fredkin's lectures, collected and organized by Bill Silver in a *6.895 Term Paper*, "Conservative Logic," MIT Dept. of Electr. Eng. Comp. Sci. (1978).

[8] Fredkin, Edward, and Toffoli, Tommaso, "Design Principles for Achieving High-Performance Submicron Digital Technologies," *Proposal to DARPA*, MIT Lab. for Comp. Sci. (1978).

[9] Hennie, C. H., *Iterative Arrays of Logical Circuits*, John Wyley and Sons (1961).

[10] Holt, Anatol, Personal communication (1979).

[11] Kinoshita, Kozo, *et al.*, "On Magnetic Bubble Circuits," *IEEE Trans. Computers* **C-25** (1976), 247-253.

[12] Landauer, Rolf, "Irreversibility and Heat Generation in the Computing Process," *IBM J.* **5** (1961), 183-191.

[13] Landauer, Rolf, "Fundamental Limitations in the Computational Process," *Tech. Rep. RC 6048*, IBM Thomas J. Watson Res. Center (1976).

[14] Mackey, G. W., *Mathematical Foundations of Quantum Mechanics*, W. A. Benjamin (1963).

[15] Meyer, J. D., and Zeigler, B. P., "On the Limits of Linearity," *Theory of Machines and Computation*, 229-242, Academic Press (1971).

[16] Moore, E. F., "Machine Models of Self-Reproduction," *Proc. Symp. Appl. Math.* (Amer. Math. Soc. **14** (1962), 17-33.

[17] Priese, Lutz, On a Simple Combinatorial Structure Sufficient for Sublying Nontrivial Self-Reproduction, *J. Cybernetics* **6** (1976), 101-137.

[18] Smith, A. R. III, "Cellular Automata Theory," *Tech. Rep. no. 2*, Stanford Electr. Lab., Stanford Univ. (1969).

[19] Sutherland, I. E., and Mead, C. M., "Microelectronics and Computer Science," *Scientific American* **237**:3 (Sept. 1977), 210-262.

[20] Toffoli, Tommaso, "Computation and Construction Universality of Reversible Cellular Automata," *J. Comput. Syst. Sci.* **15** (1977), 213-231.

[21] Toffoli, Tommaso, "Cellular Automata Mechanics" (Ph.D. Thesis), *Tech. Rep. no. 208*, Logic of Computers Group, Univ. of Michigan (1977).

[22] Toffoli, Tommaso, "The Role of the Observer in Uniform Systems," *Applied General Systems Research* (ed. G. J. Klir), 395–400 (Plenum

Press, 1978).

[23] TOFFOLI, Tommaso, "Bicontinuous Extensions of Invertible Combinatorial Functions," *Tech. Memo MIT/LCS/TM-124*, MIT Lab. for Comp. Sci. (1979) (to appear in *Math. Syst. Theory*).

[24] TURING, A. M., "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proc. London Math. Soc.*, ser. 2, 43 (1936), 544-546.

[25] VON NEUMANN, John, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Automata Studies* (edited by C. E. Shannon and J. McCarthy), 43-98, Princeton Univ. Press (1956).

[26] WIGINGTON, R. L., "A New Concept in Computing," *Proc. IRE* 47 (1961), 516-523.